

ECE 3724 – Microprocessors

Textbook: “Microprocessors: From Assembly Language to C using the PIC18Fxx2”, R. Reese.

Note: Even though this course covers the PIC18F242 microcontroller, the PhD exam questions in this course are written to be generic to any processor. You are not expected to know the PIC18 instruction set architecture; however, you are expected to be familiar with *some* microprocessor instruction set architecture and the basic instruction types/addressing modes found in any microprocessor. You are expected to be able to translate some simple high level statements written in C (numerical computations, ‘if’, loop constructs) to assembly language for a generic 8-bit processor. You are expected to be familiar with the generalities associated with the hardware interfacing topics presented below. You are not expected to be familiar with the hardware details of particular hardware modules of the PIC18F242 like the USART or A/D modules. However, you would be expected to answer general questions about how a USART operates, or the basic operation of an analog-to-digital converter (A/D).

- 1 Digital Review and Stored Program Machine Fundamentals
 - 1.1 Combinational Logic Review
 - 1.1.1 CMOS transistors, simple gates
 - 1.1.2 Building Blocks (Muxes, adders, incrementers, Memory)
 - 1.2 Sequential Logic Review
 - 1.2.1 Clocks, DFFs
 - 1.2.2 Sequential Building Blocks (registers, counters, shift registers)
 - 1.3 Stored Program Machine Components (IO, Control, Memory)
 - 1.4 Fetch/Execute Cycle, Program Counter
 - 1.5 Assembly language, Machine Code representation
 - 1.5.1 Translation of assembly code to machine code and vice-versa
 - 1.6 Number Systems
 - 1.6.1 Binary, Hex representation
 - 1.6.2 Signed Number representation (signed magnitude, 1’s complement, 2’s complement)
 - 1.6.3 Shift, Add, Subtraction operations on hex numbers
- 2 Assembly Language Programming
 - 2.1 Instruction Types
 - 2.1.1 Data Movement
 - 2.1.2 Addition/Subtraction
 - 2.1.3 Logical bitwise operations
 - 2.1.4 Shift Operations
 - 2.2 Addressing Modes
 - 2.2.1 Immediate
 - 2.2.2 Indirect (pointer addressing)
 - 2.2.3 Register
 - 2.3 Programming
 - 2.3.1 8-bit, 16-bit, 32-bit Addition/Logical operations using 8-bit assembly language instructions
 - 2.3.2 Flag usage (Carry, Negative, Overflow, Zero) in comparison operations (equality, inequality, greater-than, less-than)
 - 2.3.3 “If” statement constructs in assembly language
 - 2.3.4 Loop statement constructs (While-do, do-while) in assembly language

- 2.3.5 Subroutines (use of the stack for saving return address, parameter passing and local variable storage)
- 3 Microprocessor Architecture Fundamentals
 - 3.1 Reset Sequence
 - 3.2 Instruction execution versus clock cycles
 - 3.3 CMOS power consumption
- 4 Parallel IO Basics
 - 4.1 TTL vs CMOS logic levels
 - 4.2 Open Drain, Tri-state buffer outputs
 - 4.3 LED/Switch IO
 - 4.4 Switch Debouncing
 - 4.5 Data transfer rate computations
- 5 Asynchronous Serial IO
 - 5.1 Asynchronous IO data formats
 - 5.2 Serial data transfer rate computations
 - 5.3 Software UART vs. hardware UART operation
 - 5.4 Basic RS232 signal definitions (TX, RX, GND), voltage levels, interface ICs (MAX 232)
- 6 Interrupts
 - 6.1 Basic interrupt definition and processing/context saving
 - 6.2 Interrupt-driven LED/Switch IO
 - 6.3 Circular FIFO for interrupt driven IO
 - 6.4 Double-buffered input for interrupt-driven Stream IO
- 7 Synchronous Serial IO
 - 7.1 Serial Peripheral Interface (SPI or 3-wire interface)
 - 7.2 I²C interface
 - 7.2.1 Addressing
 - 7.2.2 Read/Write Transactions
- 8 Data Conversion
 - 8.1 Analog-to-Digital conversion basics
 - 8.1.1 Voltage to Binary representation
 - 8.1.2 Flash, Successive Approximation Architectures
 - 8.2 Digital-to-Analog conversion basics
 - 8.2.1 Binary to Voltage representation
 - 8.2.2 R-2R architecture
- 9 Timer Basics
 - 9.1 Basic Timer architecture (prescaler, postscaler, period control)
 - 9.2 Time to Timer Tics conversion; Timer Tics to Time conversion
 - 9.3 Pulse width measurement using a timer
 - 9.4 Frequency Measurement using a timer